

MIDI Lights & Slide Control

controlling slide projectors with MIDI commands

Design by A. Mylle

In the January 2001 issue we showed that it was possible to control other things with MIDI besides the usual musical instruments. This time, MIDI signals are used to control light sources. There are four adjustable triac outputs and eight relays to control, for instance, four slide projectors.



Table 1. Types of MIDI-messages.

channel voice messages	channel mode messages	system common messages	system realtime messages	system exclusive messages
note off	all sound off	midi timecode quarter frame	timing clock	system exclusive
note on	reset all controllers	song position pointer	start	sample dump
poly pressure	local control	song select	continue	device inquiry
control change	all notes off	tune request	stop	file dump
program change	omni off	eox (end of exclusive)	active sensing	...
channel pressure	omni on		system reset	midi show control
pitch bend	mono on (poly off)			midi machine control
	poly on (mono off)			...

When controlling lights, it is not easy to maintain synchronisation between light and sound when the control signals reside on different media. There exist a plethora of professional equipment that make synchronisation possible, but usually have a price tag attached that is well beyond the means of the amateur. Cheaper systems may have limited expandability and often cannot be interconnected with systems from a different brand.

Hence the idea to control (slide) projectors and relays via MIDI, where 'light events' can be manipulated with a MIDI/audio-sequencer. Synchronisation is no longer an issue, since everything is now controlled from a PC. By using MIDI as the medium, the application of the MIDI lights is now independent from any hardware or software. If you have at your disposal a MIDI-sequencer with audio capabilities, then it is possible to mix the sam-

pled audio with MIDI. And, if this sequencer possesses more than one separate MIDI-output, then it becomes possible to connect multiple light controllers.

The MIDI solution

This idea is not new. Right from the beginning, with definition V1.0 from the MMA (MIDI Manufacturers Association) were, in the chapter MSC (MIDI Show Control), definitions for protocols related to the control of, for example, lights, stroboscopes, lasers, film projectors, etc. These MSC-messages are 'system-exclusive' commands. These commands have a wide range of applications what immediately means that they are not easily created or edited using a regular sequencer. Besides, an MSC-message for the control of lights is easily 12 bytes long. If these MSC-messages are riding along the same MIDI-interfaces

as the MIDI-music then congestion on the MIDI-cable is guaranteed to arise.

More important than the load on the MIDI-cable is the problem that sysex-messages are not normally displayed in graphical form on a sequencer. There will, at best, be a list of hexadecimal numbers in the play list. To edit light events with the music is virtually impossible. There is, however, another possibility.

In **Table 1** is an overview of all the types of MIDI-messages, including a list of the most common.

Normally, commands for light control belong in the MSC-category among the 'system exclusive messages'. However, if we look at the list 'channel voice messages' to 'control change', then these messages consist of only 3 bytes: one status byte and 2 data bytes. Such a control change usually serves the purpose of altering the parameters of the sound source, such as, for example, volume, reverberation, panning, etc.

Most of the possible 128 control changes (refer to sidebar 'MIDI-bytes') are already allocated fixed functions. But, there are also a few that are free to be used for equipment

Table 2. Utilised controllers.

General purpose continuous controller #1	ctrl 16	brightness projector #1
General purpose continuous controller #2	ctrl 17	brightness projector #2
General purpose continuous controller #3	ctrl 18	brightness projector #3
General purpose continuous controller #4	ctrl 19	brightness projector #4
General purpose switch controller #5	ctrl 80	relay projector #1
General purpose switch controller #6	ctrl 81	relay projector #2
General purpose switch controller #7	ctrl 82	relay projector #3
General purpose switch controller #8	ctrl 83	relay projector #4
undefined controller	ctrl 84	extra relay #1
undefined controller	ctrl 85	extra relay #2
undefined controller	ctrl 86	extra relay #3
undefined controller	ctrl 87	extra relay #4
reset all controllers	ctrl 121	all lamps and relays off
'running status' is recognised (see text)		

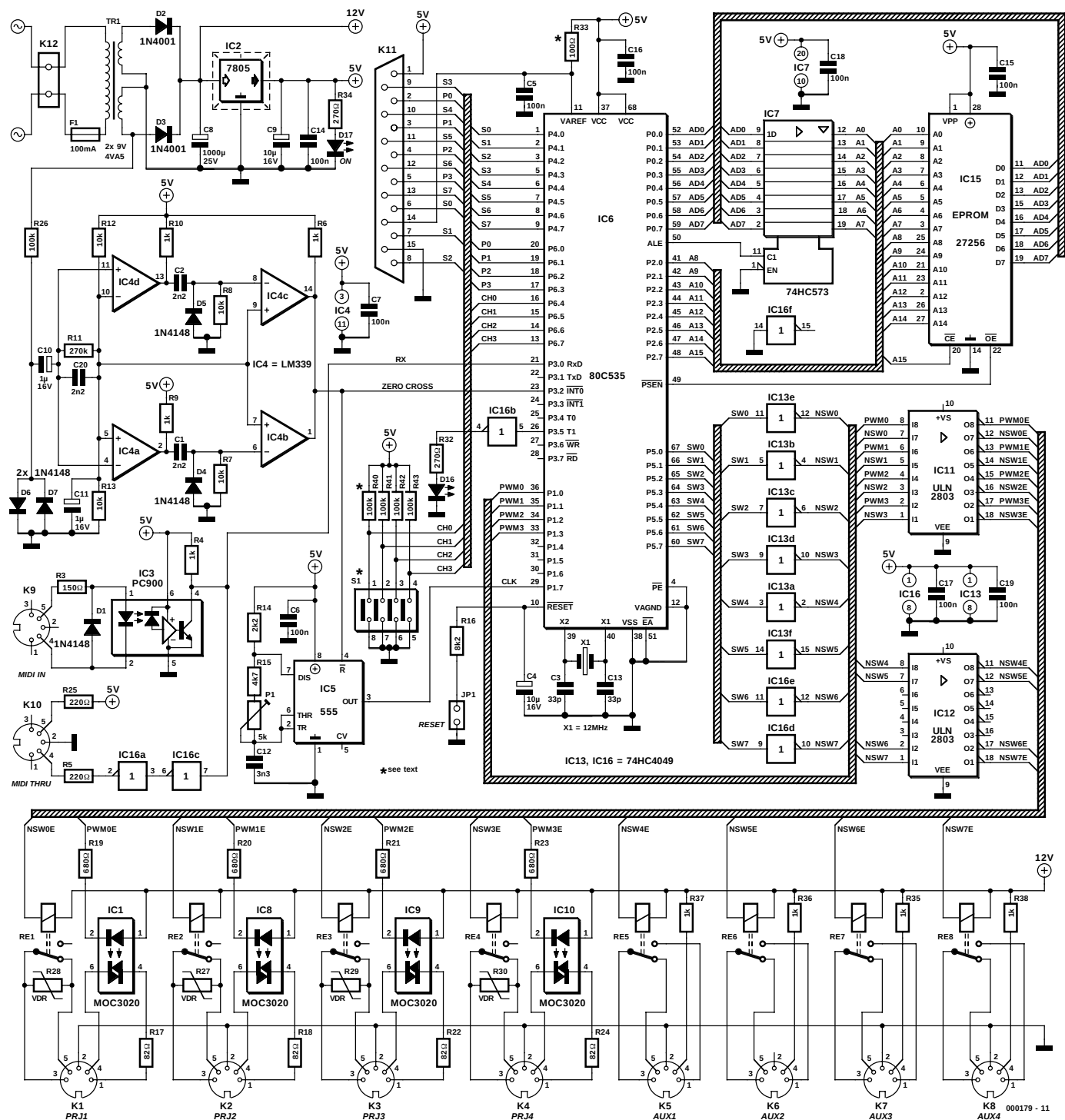


Figure 1. Schematic of the MIDI-light-control.

specific functions, for instance. If we select our light control commands from these available control changes then, suddenly, everything becomes much simpler. Lights may now be manipulated in the same way as the modulation of volume. A nice feature is that it becomes possible to program the sequencer to operate a light with the modulation wheel and switch a relay with the sus-

tain pedal. Providing the light module with a MIDI-through-connection means that it can simply be connected in series with the sound module using a standard MIDI cable. Of course, there is the prerequisite that the sound module does not respond to the controller commands used by the light module. These controller

commands have been chosen such that the risk of interference is virtually eliminated. The MIDI-light-module uses only a couple of controllers from channel 16 and since MIDI files seldom contain these specific controllers there is unlikely to be a problem. The MIDI-light-controller can drive 4 projectors and 8 relays when

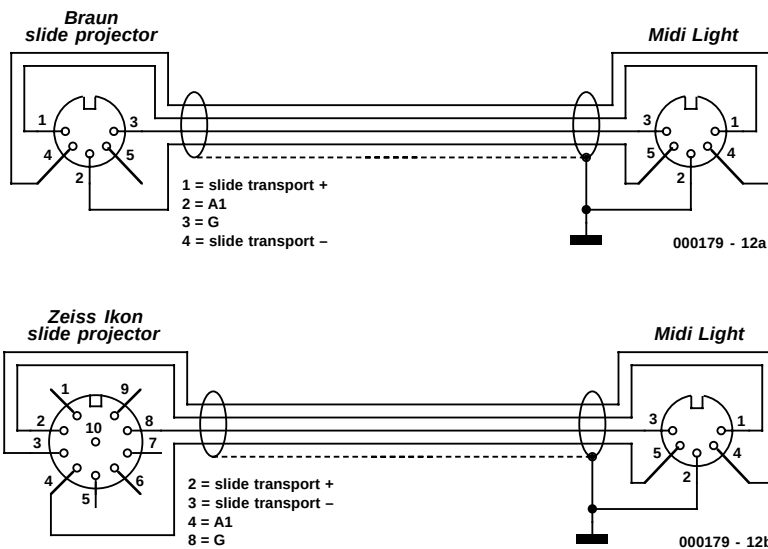


Figure 2. Connector details of the control terminals of slide projectors from Braun and Zeiss Ikon.

used with the MIDI-controllers summarised in **Table 2**.

Possibilities

Control of 4 projectors

The light intensity is controlled by the value of the controller. Every change in intensity requires an event. Value '0' means the light is off and value '127' means light is on maximum. The pre-glow is adjustable and is identical for all four channels. The lamp characteristics are linearised.

Control of 4 normally open contacts

(To advance slides, for example)

The relay is controlled in real time by

the value of the controller. Every change of state requires an event. A value of '0' means the contacts are open, a value of '127' means contacts closed. This means that the user defines the contact closure time.

Driving 4 relays

(Make and break contacts can be utilised for a variety of purposes) The relay is controlled in real time by the value of the controller. Every state change requires an event. When the value is '0' the relay is not energised, when the value is '127' the relay is energised. The DIN connector makes both the make- and break contact available together with a 12 V signal via a series resistor. This approach makes the inter-

face more versatile. Either use the contact to switch low voltage directly or use an opto-triac to switch 230 V. This last suggestion can be implemented by combining an opto-triac with a power triac and fitting this in an enclosure with built-in plug and socket. This way everything is safely separated from the 230 V mains. Never use the relay contacts to switch 230 V directly!

The hardware

The light module (**Figure 1**) is built around an 80C535-12 from Infineon (Siemens). It is a member of the 8051-family with, among other things, four pulse width modulators built around timer 2, 7 I/O ports and an 8 channel A/D-converter. The PWM section performs the phase control of the mains voltage for the 4 projector lamps.

The PWM outputs each trigger an opto-triac type MOC3020 (IC1, IC8, IC9 and IC10),

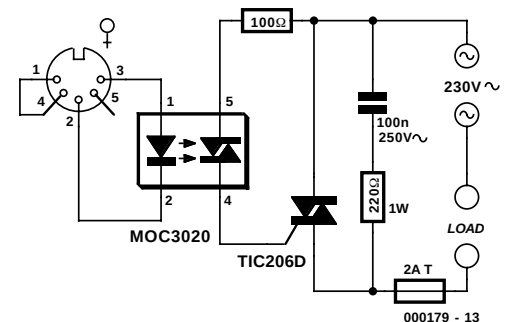


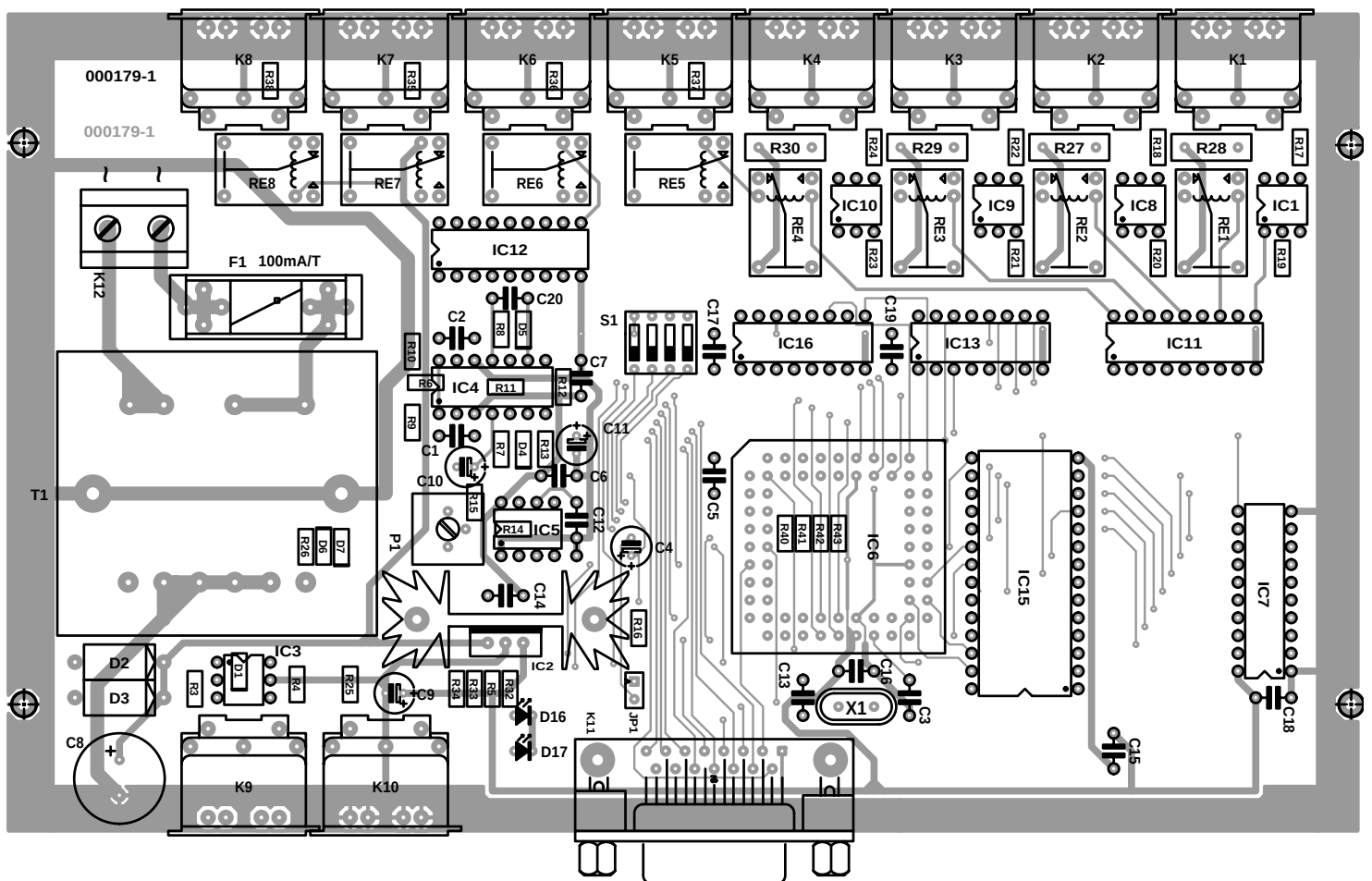
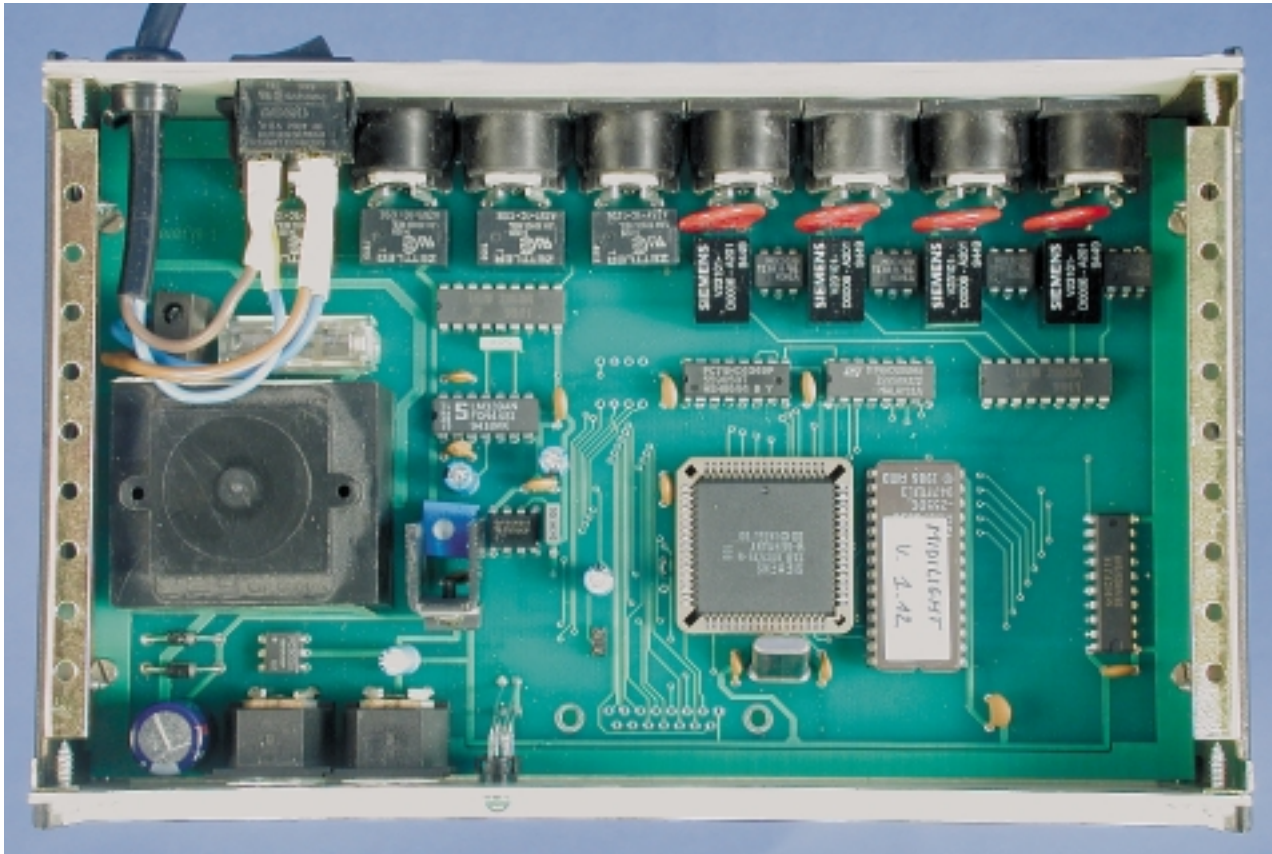
Figure 3. Schematic of a circuit to control 230 V load (max. 2 A).

which in turn fires the power triac internal to the projector. If a modification to the projector is necessary, it is limited to the fitting of a triac. If the projector has a built-in brightness control it will be sufficient to make the anode- and gate connections available on the outside. A number of common projector connections are shown in **Figure 2**.

To obtain good mains synchronisation of the pulse width modulators, a zero-crossing-detector is built around an LM339 quad-comparator (IC4) which, every 10 ms, activates interrupt 0 on pin 32. Careful attention was paid to the design of this detector to avoid interference to the light control from the 1350 Hz time code signal that rides on top of the 50 Hz mains voltage. The AC signal is tapped from the (highest available) voltage on the secondary of the mains transformer. On the outputs of the first two comparators, alternating between them, appears a falling

Running status

MIDI-commands (such as 'control change') consist always of a single status byte followed by one or more data bytes. The status byte defines the type of command that is to follow, while the data bytes contain the actual information. In the case of the control change, for example, the status byte is followed by 2 more data bytes. The first contains the controller number and the second contains the controller value. When the next MIDI-message is intended for the same MIDI-channel, it is not necessary to repeat the status byte. This status byte would have been identical to the previous one. Such successive MIDI-commands of the same type and for the same channel require only a single status byte, sent at the beginning of the set of messages. A status byte is only transmitted if it differs from the previous one. This entire mechanism is called 'running status'. Some older MIDI equipment is unable to handle the running status. The better sequencers permit this mode to be switched off.



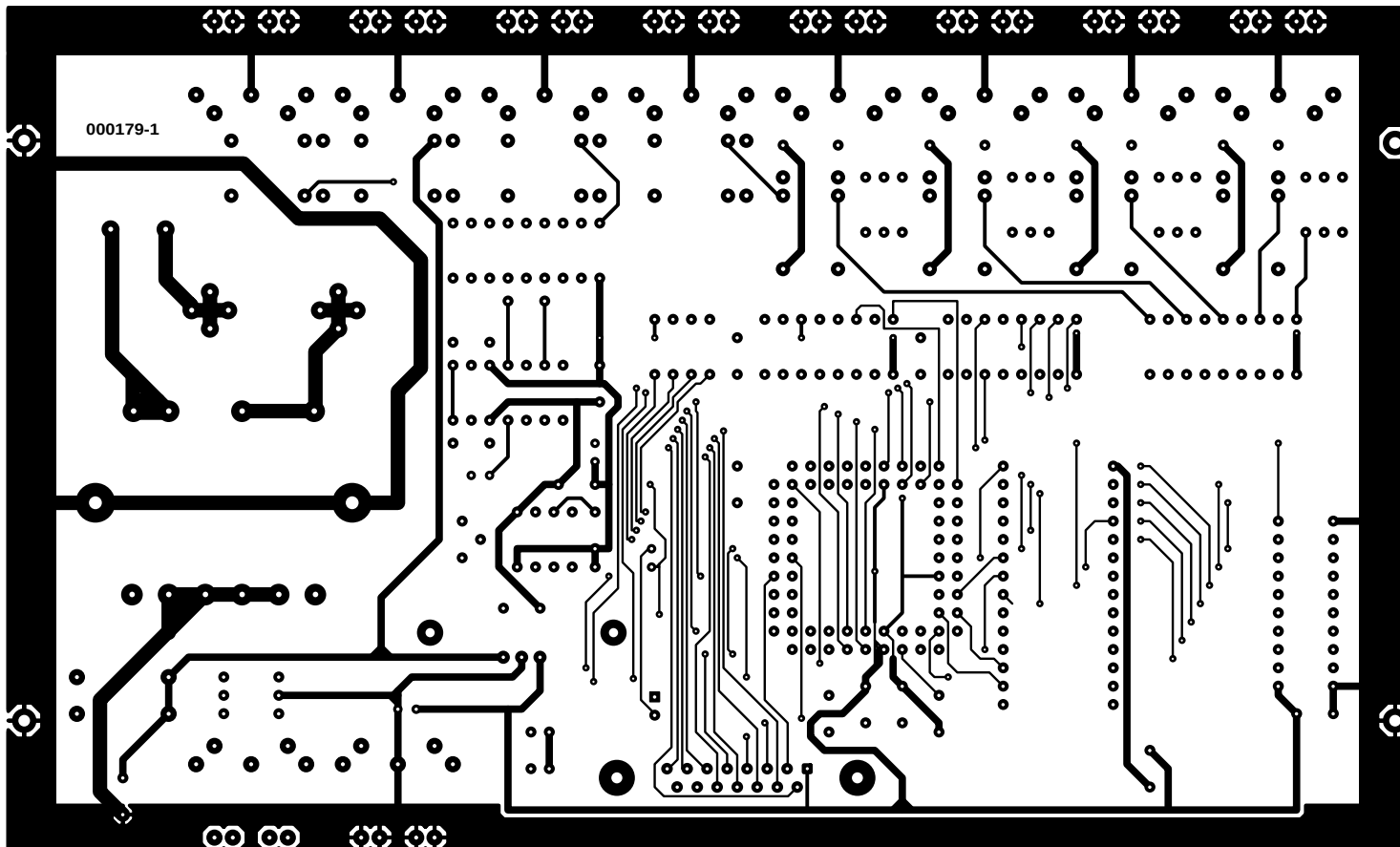
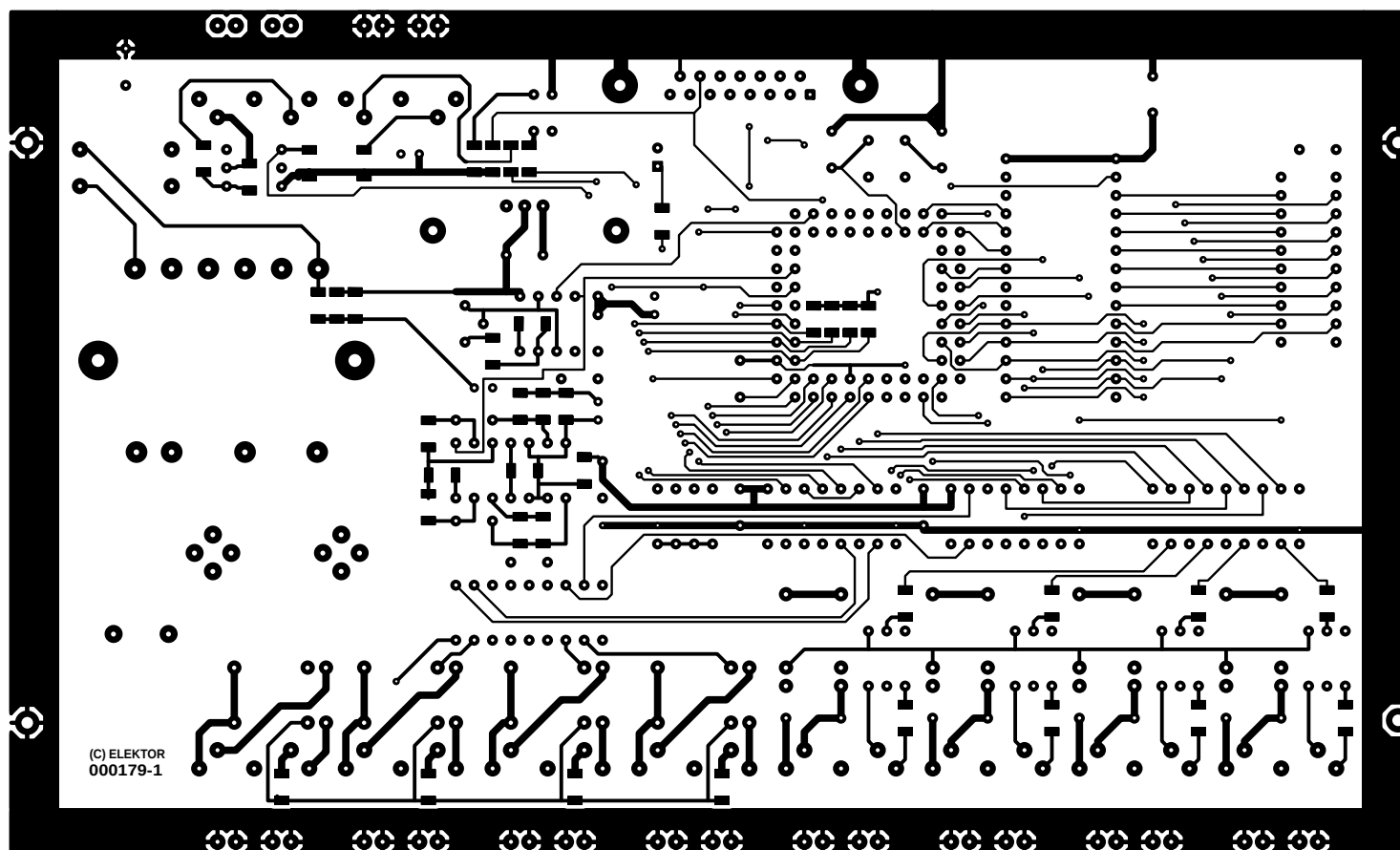


Figure 4. The PCB is double sided. It also contains the mains transformer and all connectors.

edge at the instance of each zero-crossing. Each falling edge is converted to a short pulse of a few microseconds duration using an RC-network and a second pair of comparators. This pulse is sufficiently long to be detected by the interrupt routine. As a consequence, this routine is executed every 10 ms.

Because the zero-crossing is detected at the secondary winding of the transformer in the module itself and not the projector, the actual zero-crossing of the lamp may not coincide exactly with the pulse from the detector. You will not notice this in practice, provided both the module and the projector are powered from the same phase.

The PWM-clock of about 28 kHz is generated by IC5, a 555. This signal is reset at every zero-crossing. This is necessary to ensure that every half mains cycle is started with a whole pulse. If this were not done, there would be small fluctuations in the brightness of the lamp because the PWM-clock is not an integer multiple of the mains frequency.

The MIDI-IN-interface is, as prescribed in the MIDI standard, not much more than an opto-coupler PC900 (IC3) connected to the serial port of the microcontroller. The MIDI-input is also immediately looped to the MIDI-output, which enables the light module to be connected in series with existing equipment. LED D16 lights up briefly when the module has received signals that control the projectors or relays.

Relays Re1-Re8 are driven from port 5 via buffers IC13 and IC16 and relay drivers IC11 and IC12.

The contacts of relays Re1-Re4 are suppressed with VDRs (R27-R30).

Latch IC7 filters, with the aid of the ALE signal, the least significant byte of the address bus from the multiplexed address/data bus for use by the EEPROM.

The software

This comprises 3 parts:

The zero-crossing detector triggers the INTO routine, which performs the phase control by reading the CC-registers.

The serial input places all received MIDI-bytes in a buffer, 128 bytes long.

The main program reads the bytes from the buffer and processes the relevant commands. It detects the running status, drives the relays and refreshes the CC-registers.

Phase control is obtained as follows: The pulse from the zero-crossing detector triggers IRQ0 every 10 ms – one half mains cycle – which sets timer 2 to a value of 65535 (FFFFh). The first PWM clock pulse causes

MIDI-bytes

With MIDI, the MSB defines if it is a status byte (MSB='1') or a data byte (MSB='0'). This leaves 7 bit values for the data bytes. With control changes, the first byte contains the number of the controller and the second contains the actual controller value. If a higher resolution is required, such as for 'pitch bend', then an extra byte is sent for the controller value, or as with some controllers, a separate MSB- and LSB-controller is used. There are a maximum of 128 controllers that may be operated. Fortunately, the existence of MIDI-standards such as GM, GS and XG ensures that, among other things, a certain controller number on different pieces of equipment has the same function. In this way MIDI-files sound almost identical on sound sources from different manufacturers.

the timer to overflow which then counts up from zero. As a consequence of this overflow, all PWM outputs are set to '0'. All this occurs

immediately after the zero-crossing. The counter continues to count at the rate of the PWM clock, which generates 256 pulses in 10 ms (see

COMPONENTS LIST

Resistors:

(All resistors SMD case # 1206)

R1,R2,R31 = not fitted

R3 = 150Ω

R4,R6,R9,R10,R35-R38 = 1kΩ

R5,R25 = 220Ω

R7,R8,R12,R13 = 10kΩ

R11 = 270kΩ

R14 = 2kΩ

R15 = 4kΩ

R16 = 8kΩ

R17,R18,R22,R24 = 82Ω

R19-R21,R23 = 680Ω

R26,R40...R43* = 100kΩ

R27-R30 = VDR 48-96 V, e.g., Siemens S10K50 or S07K50

R32,R34 = 270 Ω

R33 = 100Ω*

P1 = 5kΩ preset, 'ordinary' vertical model or Bourns type 3386

Capacitors:

C1,C2,C20 = 2nF

C3,C13 = 33pF

C4,C9 = 10μF 16V radial

C5,C6,C7,C14-C19 = 100nF, lead pitch 5mm

C8 = 1000μF 25V radial

C10,C11 = 1μF 16V radial

C12 = 3nF

C14 = 100pF

Semiconductors:

D1,D4-D7 = LL4148 (SMD)

D2,D3 = 1N4001

D16 = LED, green

D17 = LED, yellow

IC1,IC8,IC9,IC10 = MOC3020 (QT, formerly Motorola: www.qtopto.com)

IC2 = 7805

IC3 = PC900 (Sharp)

IC4 = LM339

IC5 = 555

IC6 = SAB80C535-N

IC7 = 74HC573

IC11,IC12 = ULN2803

IC13,IC16 = 74HC4049

IC15 = 27C256, programmed, order code **000179-21**

Miscellaneous:

K1-K10 = 5-way DIN socket, PCB mount, pins arranged at 180°

K11 = 15-way sub-D socket (female), PCB mount, angled pins*

K12 = 2-way PCB terminal block, lead pitch 7.5 mm

X1 = 12 MHz quartz crystal

F1 = fuse, 100 mA (time lag), with PCB mount holder

S1 = 4-way DIP switch

T1 = mains transformer, sec. 2 x 9 V / 4.5VA (e.g., Monacor/Monarch VTR4209)

JP1 = 2-way pinheader

68-pin PLCC socket

Heatsink for IC2, e.g., SK104-50 (Fischer, Dau Components)

Re1-Re8 = 12-V relay (e.g., Zettler AZ5Y-1C-12DE or Siemens V23101-D6-A201)

Enclosure type LC650 (Eurodis, Bolton)

PCB, order code **000179-1** (see Readers Services page and EE website)

Disk, contains sourcecode and binary file, order code **000179-11** (see Readers Services page and EE website)

* optional, for extensions

Generating Controller Events using Cubasis AV

It is relatively easy, using the program Cubasis from Steinberg to generate controller commands. Start by setting the time base. If you start with an existing MIDI-file than this is already fixed, of course. But if you use only audio, then choose this yourself. The time axis is divided in bars. The number of bars that are played per second are: $\text{Tempo} \times \text{measure} / 60$. Example: 120 bpm (beats per minute) and 2/4 equals one bar per second. Pay attention, the first bar is '1'. At bar number 10, 9 seconds have elapsed. It is easiest to create 'parts' for each series of controllers. For example, make parts for long fade in, short fade in, fade out, change, etc. Every part has a name. Each part contains the controller events.

A controller event is created as follows:

Define the length of a part and set markers at the desired beginning and end. Select the active track on the left and set this to MIDI-channel 16 of the MIDI-out interface.

Double click on the empty space between the markers at the height of the track. There appears an empty part.

Select this part.

Click 'Edit – List' and in the pull down list 'ins. control change'.

On the right of the screen, select the pencil using the right mouse button and mark a first event at the beginning of the part.

Change the list 'val. 1' to the desired controller number. This is '15' for the first channel. In the status field appears the name of the controller 'General Purpose 1'. The brightness 'val. 2' is from 0 (off) to 127 (max).

Click 'Edit – Edit'. At bottom left there is now controller number 15. Don't click it, because the pull down menu does not contain all 127 controllers. For other controllers it only works when starting from a selected controller in 'List Edit'.

In the bottom part of the screen, click either the cross or the pencil with the right mouse button. With the alt key held down, draw new events by dragging the starting value of the end value. The resolution can be changed using the 'snap'-parameter.

Parts for advancing slides consist of first sending a value of 127 followed by 0. The time between these events can be defined and on some projectors this is used to go forwards or backwards.

Once you have a 'library' of a number of basic parts, most of the hard work has been done, you can now simply copy and paste.

below). Every PWM-output has a 'capture&compare'-register. The register for each channel contains a value that corresponds to the desired phase angle. When the counter value of timer 2 is equal to the value in the CC-register, the corresponding PWM-output becomes '1' and triggers the triac. By varying the value in the CC registers, the phase angle of each channel can be controlled from lamp off (CC equals FFFFh) to lamp fully on (CC equals zero).

The CCs are 16-bit registers while the MIDI-controls are only 128 values (7 bits). We vary only the LSB (MSB=FFFh) such that the values 0-127 from the MIDI-controller are translated to 255...0 using a table.

Making the table slightly non-linear allows the lamp characteristic to be linearised. When adjusting the brightness of the lamp from zero to maximum it appears that all the steps are equally spaced across the whole range.

Setting the PWM-clock to slightly more than 256 pulses, for example 280 pulses per 10 ms (=28 kHz), causes the counter to trip a little before the next zero crossing, when set to minimum brightness. As a consequence, the lamp is not completely off at the minimum setting. This pre-glow ensures that the lamp reacts faster when fading in. Because this adjustment is part of the PWM-clock, it is the same for all lamps.

Except for reloading timer 2 during the interrupt routine for the phase control, the involvement of the microcontroller is limited to refreshing the CC-registers. It is imperative that this happens at the zero crossing. If this is not the case then the lamp may flicker because the new value may be smaller than the current counter value.

Incoming MIDI-bytes are put into a 128 byte long buffer by the serial interrupt routine. The main program processes the bytes in the buffer.

Construction and Adjustment

A double-sided PCB was designed for this circuit. It includes the mains transformer and all connectors (**Figure 3**).

When purchasing the PLCC68 socket for the microcontroller, check that it suits the PCB since multiple pin configurations exist. The construction should otherwise be straightforward. It is best if you mount the components in the following order: passive components, the diodes, the IC sockets, the relay and finally the connectors. Mount the ICs only after you have checked the power supply voltages. With the exception of IC1 and IC3, it is possible to fit the other ICs without a socket. The PCB layout allows for several types of mains transformer. Make sure that the primary side of the transformer is on the same side as the fuse.

Adjust P1 to the centre position and connect the projectors to K2-K5. In the event of a three-phase 230 V mains, ensure that all the projectors are connected to the same phase. As a check, LED D16 should light briefly after switching the power supply on (this also happens after a manual reset, using jumper J1). Turn P1 until the lamps just show a dim red glow.

This concludes the adjustments. You can now send MIDI signals to control the module. Of course, the intricate details of how that is done depend on the sequencer you use. The LED will light briefly every time a control message has been received.

(000179-1)

A few useful MIDI-links

The MIDI Farm Internet:

<http://www.midifarm.com/info/>

MMA home page at www.midi.org:

<http://www.midi.org/>

Harmony Central: MIDI Tools and Resources:

<http://www.harmony-central.com/MIDI/>